**Отчет по лабораторной работе №13** по курсу \_\_Фундаментальная информатика\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы М80-101Б-21 Постнов Александр Вячеславович, № по списку 17

Контакты www, e-mail:

Работа выполнена: «» 202 1г.

Преподаватель: каф. 806 \_\_\_\_\_Титов В.К.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_2021\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Тема:** Множества, файлы с расширением .h\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Цель работы:** составить и отладить программу на СИ на заданную тему. \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Задание** (*вариант: 17*)**:** Выписать соседние слова с непересекающимся набором гласных.
2. **Оборудование(лабораторное)**:

ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,

НМД - ГБ, терминал- адрес -, принтер -

Другие устройства -

*Оборудование ПЭВМ студента, если использовалось:*

Процессор AMD Ryzen 5 4500U, с ОП 8 ГБ

Другие устройства -

1. **Программное обеспечение:**

Операционная система семейства -, наименование - версия - интерпретатор команд - версия

Система программирования - версия -

Редактор текстов - версия -

Утилиты операционной системы -

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства GNU/Linux, наименование Pop!\_OS версия 21.04

интерпретатор команд GNOME Terminal версия 3.38.2

Система программирования \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Редактор текстов emacs версия 3.27.20

Утилиты операционной системы cat

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

1. **Идея, метод, алгоритм**  решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Словесный алгоритм:

1) Создам файл с расширением .h(собственную библиотеку), в котором будет реализовано множество и операции над множествами:

Множество будет реализовано через массив, который будет содержать 52 элемента(для строчных и заглавных букв латинского алфавита).

Но в процедуре InSet добавление букв будет будет одинаково и для строчных, и для заглавных букв, так как в моей задаче нет разницы строчная гласная или заглавная.

2) Алгоритм главной программы(main.c):

Массив Sv – множество гласных букв латинского алфавита

Массив Svo – множество гласных букв предыдущего слова

Массив Svn – множество гласных букв текущего слова

Массив oldWord – предыдущее слово

Массив nowWord – текущее слово

В каждой итерации цикла:

1) будет обнуляться текущее слово, множество гласных букв текущего слова

2) считываем символ, до тех пор, пока не встретим букву.

3) если встретили букву – значит, это начало нового слова, запускаем цикл до тех пор, пока считываемый символ – это буква.

4) Проверяем, что два соседних слова существуют(на текущей итерации цикла), с помощью проверки первых символов массивов oldWord и nowWord, если они буквы, то эти слова есть.

5) Пусть S\_inter – это пересечение гласных букв текущего слова и предыдущего слова, если оно пустое, то выписываем два этих слова через пробел на новой строке.

6) Текущее слово и текущее его множество гласных букв становятся предыдущими(если текущее слово существует).

**7. Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

Библиотека Set.h

// file Set.h

// --------------------------------------------------

#define N 26

char A = 'A';

char Z = 'Z';

char a = 'a';

char z = 'z';

char Alph[2 \* N]; // alphabet: small and capital letters

int Letter(char c)

{

return c >= a && c <= z || c >= A && c <= Z;

}

void CreateSet(char W[], char S[]) //создание множества

{

char c;

int i;

for (i = 0; i < N; i++)

S[i] = S[N + i] = 0;

for (i = 0; c = W[i]; i++)

if (c >= a && c <= z)

S[c - a] = 1;

else if (c >= A && c <= Z)

S[N + c - A] = 1;

}

void DisplaySet(char S[])

{

int i;

Alph[0] = a;

Alph[N] = A;

Alph[2 \* N] = 0; // alphabet fills up

for (i = 1; i < N; i++)

{

Alph[i] = Alph[i - 1] + 1;

Alph[i + N] = Alph[i + N - 1] + 1;

} // --"--

printf("\n%s\n", Alph); // and prints

for (i = 0; i < 2 \* N; i++)

printf("%1d", S[i]);

printf("\n"); // set prints

}

int EqualSet(char S1[], char S2[])

{

for (int i = 0; i < 2 \* N; i++)

if (S1[i] != S2[i])

return 0;

return 1;

}

void UnionSet(char S1[], char S2[], char S3[])

{

for (int i = 0; i < 2 \* N; i++)

S3[i] = S1[i] || S2[i];

}

void IntersSet(char S1[], char S2[], char S3[])

{

for (int i = 0; i < 2 \* N; i++)

S3[i] = S1[i] && S2[i];

}

int IncludeSet(char S1[], char S2[])

{

for (int i = 0; i < 2 \* N; i++)

if (S1[i] == 1 && S2[i] == 0)

return 0;

return 1;

}

int EmptySet(char S[]) // Пустое ли множество

{

for (int i = 0; i < 2 \* N; i++)

if (S[i] == 1)

return 0;

return 1;

}

void InSet(char c, char S[]) //добавить элемент в множество

{

if (Letter(c))

if (c >= a && c <= z)

S[c - a] = 1;

else

S[c - A] = 1;

}

int FindinSet(char c, char S[]) //нахождение элемента в множестве

{

if (Letter(c)){

if (c >= a && c <= z){

return S[c - a] == 1;

}

else {

return S[c - A] == 1;

}

}

else

{

return 0;

}

}

Главная программа main.c:

#include <stdio.h>

#include "Set.h"

int main(){

char Vowels[] = "AEIOUYaeiouy";

char Sv[N \* 2];

char Svo[N \* 2]; //множество гласных для старого слова

char Svn[N \* 2]; //множество гласных для текущего слова

for (int i = 0; i < 2 \* N; i++){

Svo[i] = 0;

Svn[i] = 0;

}

char oldWord[20]; //старое слово

char nowWord[20]; //текущее слово

int i = 0;

char c;

CreateSet(Vowels, Sv);

while (i != EOF){

int counter = 0;

for (int i = 0; i < 20; i++) nowWord[i] = 0;

for (int i = 0; i < 2 \* N; i++) Svn[i] = 0;

i = getchar();

c = i;

if (Letter(c)){ //начало слова

nowWord[counter++] = c;

if (FindinSet(c, Sv)){

InSet(c, Svn);

}

while(i != EOF && Letter(c)){ //до тех пор, пока текущий символ - это буква(до конца слова)

i = getchar();

c = i;

if (Letter(c)){

nowWord[counter++] = c;

if (FindinSet(c, Sv)){

//printf("%c \n", c);

InSet(c, Svn);

}

}

}

}

if (Letter(nowWord[0]) && Letter(oldWord[0])){ //проверка, что 2 соседних слова существуют

char S\_inter[N \* 2];

IntersSet(Svn, Svo, S\_inter); //S\_inter - пересечение множества гласных букв текущего слова и старого слова

if (EmptySet(S\_inter)){ //если оно пустое выписываем эти два слова

for (int i = 0; i < 20; i++){

if (Letter(oldWord[i])){

printf("%c", oldWord[i]);

}

else {

break;

}

}

printf(" ");

for (int i = 0; i < 20; i++){

if (Letter(nowWord[i])){

printf("%c", nowWord[i]);

}

else {

break;

}

}

printf("\n");

}

}

if (Letter(nowWord[0])){ //если текущее слово существует

for (int i = 0; i < 2 \* N; i++){ //текущее множество гласных букв становится старым

Svo[i] = Svn[i];

}

for (int i = 0; i < 20; i++){ //текущее слово становится старым

oldWord[i] = nowWord[i];

}

}

}

return 0;

}

Тесты:

1)Hello, i am giant. New hero one word two word gt rr ggg

Ответ:

Hello i

i am

giant New

word gt

gt rr

rr ggg

2)One, two, three I go looking

Ответ:

two three

three I

I go

Комментарий к тесту: проверка, что строчная гласная и заглавная гласная не отличаются.

*Допущен к выполнению работы.*  **Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8. Распечатка протокола**  (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

[alex@alex 13(?)]$ cat head.txt

-----------------------------------------------------

| Лабораторная работа №13 |

| Множества. |

| Выполнил: студент группы М8О-101Б-21 |

| Постнов Александр Вячеславович |

-----------------------------------------------------

[alex@alex 13(?)]$ cat Set.h

// file Set.h

// --------------------------------------------------

#define N 26

char A = 'A';

char Z = 'Z';

char a = 'a';

char z = 'z';

char Alph[2 \* N]; // alphabet: small and capital letters

int Letter(char c)

{

return c >= a && c <= z || c >= A && c <= Z;

}

void CreateSet(char W[], char S[]) //создание множества

{

char c;

int i;

for (i = 0; i < N; i++)

S[i] = S[N + i] = 0;

for (i = 0; c = W[i]; i++)

if (c >= a && c <= z)

S[c - a] = 1;

else if (c >= A && c <= Z)

S[N + c - A] = 1;

}

void DisplaySet(char S[])

{

int i;

Alph[0] = a;

Alph[N] = A;

Alph[2 \* N] = 0; // alphabet fills up

for (i = 1; i < N; i++)

{

Alph[i] = Alph[i - 1] + 1;

Alph[i + N] = Alph[i + N - 1] + 1;

} // --"--

printf("\n%s\n", Alph); // and prints

for (i = 0; i < 2 \* N; i++)

printf("%1d", S[i]);

printf("\n"); // set prints

}

int EqualSet(char S1[], char S2[])

{

for (int i = 0; i < 2 \* N; i++)

if (S1[i] != S2[i])

return 0;

return 1;

}

void UnionSet(char S1[], char S2[], char S3[])

{

for (int i = 0; i < 2 \* N; i++)

S3[i] = S1[i] || S2[i];

}

void IntersSet(char S1[], char S2[], char S3[])

{

for (int i = 0; i < 2 \* N; i++)

S3[i] = S1[i] && S2[i];

}

int IncludeSet(char S1[], char S2[])

{

for (int i = 0; i < 2 \* N; i++)

if (S1[i] == 1 && S2[i] == 0)

return 0;

return 1;

}

int EmptySet(char S[]) // Пустое ли множество

{

for (int i = 0; i < 2 \* N; i++)

if (S[i] == 1)

return 0;

return 1;

}

void InSet(char c, char S[]) //добавить элемент в множество

{

if (Letter(c))

if (c >= a && c <= z)

S[c - a] = 1;

else

S[c - A] = 1;

}

int FindinSet(char c, char S[]) //нахождение элемента в множестве

{

if (Letter(c)){

if (c >= a && c <= z){

return S[c - a] == 1;

}

else {

return S[c - A] == 1;

}

}

else

{

return 0;

}

}

[alex@alex 13(?)]$ cat main.c

#include <stdio.h>

#include "Set.h"

int main(){

char Vowels[] = "AEIOUYaeiouy";

char Sv[N \* 2];

char Svo[N \* 2]; //множество гласных для старого слова

char Svn[N \* 2]; //множество гласных для текущего слова

for (int i = 0; i < 2 \* N; i++){

Svo[i] = 0;

Svn[i] = 0;

}

char oldWord[20]; //старое слово

char nowWord[20]; //текущее слово

int i = 0;

char c;

CreateSet(Vowels, Sv);

while (i != EOF){

int counter = 0;

for (int i = 0; i < 20; i++) nowWord[i] = 0;

for (int i = 0; i < 2 \* N; i++) Svn[i] = 0;

i = getchar();

c = i;

if (Letter(c)){ //начало слова

nowWord[counter++] = c;

if (FindinSet(c, Sv)){

InSet(c, Svn);

}

while(i != EOF && Letter(c)){ //до тех пор, пока текущий символ - это буква(до конца слова)

i = getchar();

c = i;

if (Letter(c)){

nowWord[counter++] = c;

if (FindinSet(c, Sv)){

//printf("%c \n", c);

InSet(c, Svn);

}

}

}

}

if (Letter(nowWord[0]) && Letter(oldWord[0])){ //проверка, что 2 соседних слова существуют

char S\_inter[N \* 2];

IntersSet(Svn, Svo, S\_inter); //S\_inter - пересечение множества гласных букв текущего слова и старого слова

if (EmptySet(S\_inter)){ //если оно пустое выписываем эти два слова

for (int i = 0; i < 20; i++){

if (Letter(oldWord[i])){

printf("%c", oldWord[i]);

}

else {

break;

}

}

printf(" ");

for (int i = 0; i < 20; i++){

if (Letter(nowWord[i])){

printf("%c", nowWord[i]);

}

else {

break;

}

}

printf("\n");

}

}

if (Letter(nowWord[0])){ //если текущее слово существует

for (int i = 0; i < 2 \* N; i++){ //текущее множество гласных букв становится старым

Svo[i] = Svn[i];

}

for (int i = 0; i < 20; i++){ //текущее слово становится старым

oldWord[i] = nowWord[i];

}

}

}

return 0;

}[alex@alex 13(?)]$ g++ main.c

[alex@alex 13(?)]$ cat test1.txt

Hello, i am giant. New hero one word two word gt rr ggg

[alex@alex 13(?)]$ ./a.out <test1.txt

Hello i

i am

giant New

word gt

gt rr

rr ggg

[alex@alex 13(?)]$ cat test2.txt

One, two, three I go looking

[alex@alex 13(?)]$ ./a.out <test2.txt

two three

three I

I go

**9. Дневник отладки** должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. или дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
| 1 | дом | 27.11 | 11:00 | Программа не скомпилировалась из-за синтаксической ошибки | Добавил ; в конец 15 строки |  |
|  |  |  |  |  |  |  |

**10. Замечания автора :**

11.**Выводы**

\_\_В ходе лабораторной работе я составил и отладил программу на СИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Недочёты при выполнении задания могут быть устранены следующим образом: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента \_Постнов\_\_\_\_\_\_\_\_\_\_\_\_\_\_